TAMPEREEN
AMMATTIKORKEAKOULU

ZABBIX JÄRJESTELMÄ VALVONTA

Sovellus valvonta

Peter Jokinen

Opinnäytetyö
Toukokuu 2018
Tieto- ja viestintätekniikka
Ohjelmistotekniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tieto- ja viestintätekniikka
Ohjelmistotekniikka

Peter Jokinen:
Zabbix järjestelmä valvonta
Sovellus valvonta

Opinnäytetyö 19 sivua, joista liitteitä 1 sivua
Toukokuu 2018

Tässä opinnäytetyössä perehdytään Zabbix järjestelmä valvonta ohjelmistoon, joka on vapaasti käytettävissä oleva avoimen lähdekoodin ohjelmisto. Oppinnäytetyö liittyy työtehtäviini Digia Oyj:llä.

Tässä projektissa keskitytään keskitettyyn sovellustason valvontaan, käyttäen Zabbixin sovellustason tunnistusta sekä käyttäjä parametrejä. Tavoitteena se että saadaan yksi keskitetty valvonta järjestelmä, johon saadaan näkyviin sekä järjestelmien toiminta sekä niiden sisäiset sovellukset.

Projektissa toteutetaan valvonta IBM Integration Bus – ohjelmistolle, mutta käytettyjä metodeja voidaan soveltaa lähes minkä tahansa sovelluksen valvontaan. Tässä projektissa käytettyjä teknologioita ovat mm. Python, MQTT sekä Unix Shell Bash.

Tämä projekti on avointa lähdekoodia, jonka GitHub sivulle löytyy linkki liitteessä 1.

Asiasanat: avoin lähdekoodi, sovellustason tunnistus, käyttäjä parametri, IBM Integration Bus (IIB), MQTT, Unix Shell

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Information and Communication Technology
Software techniques

Peter Jokinen:
Zabbix system monitoring
Software monitoring

Bachelor's thesis 19 pages, appendices 1 pages
May 2018

This thesis is related to one of my work assignment at Digia Oyj, where I, as a System Developer, create system monitoring solutions with the enterprise-class open source monitoring software Zabbix.

This project focuses on Zabbix Low-level discovery and user parameters, and goes through the process of making a monitoring solution, for nearly any software. The goal being to have only one software (Zabbix) with all the systems (both hardware and software) necessary to be monitored in one place.

In this case we will be creating a monitoring solution for IBM Integration Bus, but the same principals can be used for other cases as well. Key technologies in this solution are Python, MQTT and Unix Shell Bash.

All code regarding this project is released as open source software. A link to the project GitHub page can be found in appendix 1.

Key words: open source, low-level discovery, user parameter, IBM Integration Bus (IIB), MQTT, unix shell

SISÄLLYS

LYHENTEET JA TERMIT

avoin lähdekoodi	(eng. open source) koodi on kaikkien saatavilla ja nähtävillä, usein myös vapaasti käytettävissä
Käyttäjä parametri	(eng. user parameter) Zabbix:ssa käyttäjän asettamia parametrejä, esim. järjestelmä komentoja tai koodi tiedostoja, joilla voidaan kerätä tietoja
Sovellustason tunnistus	(eng. Low-level discovery) Zabbix sovellustason tunnistus, käytetään sovellusten automaattiseen tunnistamiseen järjestelmässä jolloin uusien järjestelmien valvonta alkaa automaattisesti
IBM Integration Bus (IIB)	integraatio ohjelmisto, väliohjelmisto
MQTT	Message Queuing Telemetry Transport – viestintä protokolla
Unix Shell	Komentorivi tulkki/ käyttöliittymä
integraatio	useamman järjestelmän yhdistäminen yhdeksi kokonaisuudeksi
väliohjelmisto	ohjelmisto joka varmistaa, että kaksi ohjelmistoa pystyvät toimimaan keskenään oikein (eng. middleware)
SNMP	Simple Network Management Protocol – tietoliikenne protokolla
IPMI	Intelligent Platform Management Interface – rajapinta järjestelmän etähallintaan sekä valvontaan
SSH	Secure Shell – salattu viestintä protokolla
Internet of Things (IoT)	esineiden internet, joka päiväisiä laitteita jotka on kytketty internettiin

1 JOHDANTO

Tehtäviini Digia Oyj:llä kuuluu väliohjelmistojen toteuttaminen eri valvontaratkaisuihin, ja osana tätä käytetään avoimen lähdekoodin valvonta ohjelmistoa nimeltä Zabbix. Tässä opinnäytettyössä on tarkoitus käydä läpi, miten Zabbix:ssa voidaan toteuttaa lähes minkä tahansa järjestelmän ja sen sovellusten valvonta (edellytyksenä että sovelluksen tapahtumiin päästään käsiksi myös sovelluksen ulkopuolelta), joka on myös mahdollisimman vaivaton ylläpitää tai laajentaa.

Esimerkkinä tässä työssä toteutetaan valvonta IBM Integration Bus:lle (jatkossa IIB), mutta samaa menetelmää voidaan soveltaa muillekkin ohjelmistoille. Zabbix tarjoaa valmiit työkalut perus käyttöjärjestelmä valvontaan (kuten prosessorin-, muistin- ja levynkäyttöä), mutta itse järjestelmän ohjelmistojen valvontaan tarvitaan Zabbix käyttäjä parametreja sekä sovellustason tunnistusta. Käyttäjä parametrit mahdollistavat oman koodin ajamisen ja tulosten seuraamiseen Zabbix:lla, ja sovellustason tunnistuksella uusien järjestelmien lisääminen saadaan mahdollisimman vaivattomaksi.

Projekti toteutetaan avoimena lähdekoodina jonka GitHub sivulle linkki liitteessä 1.

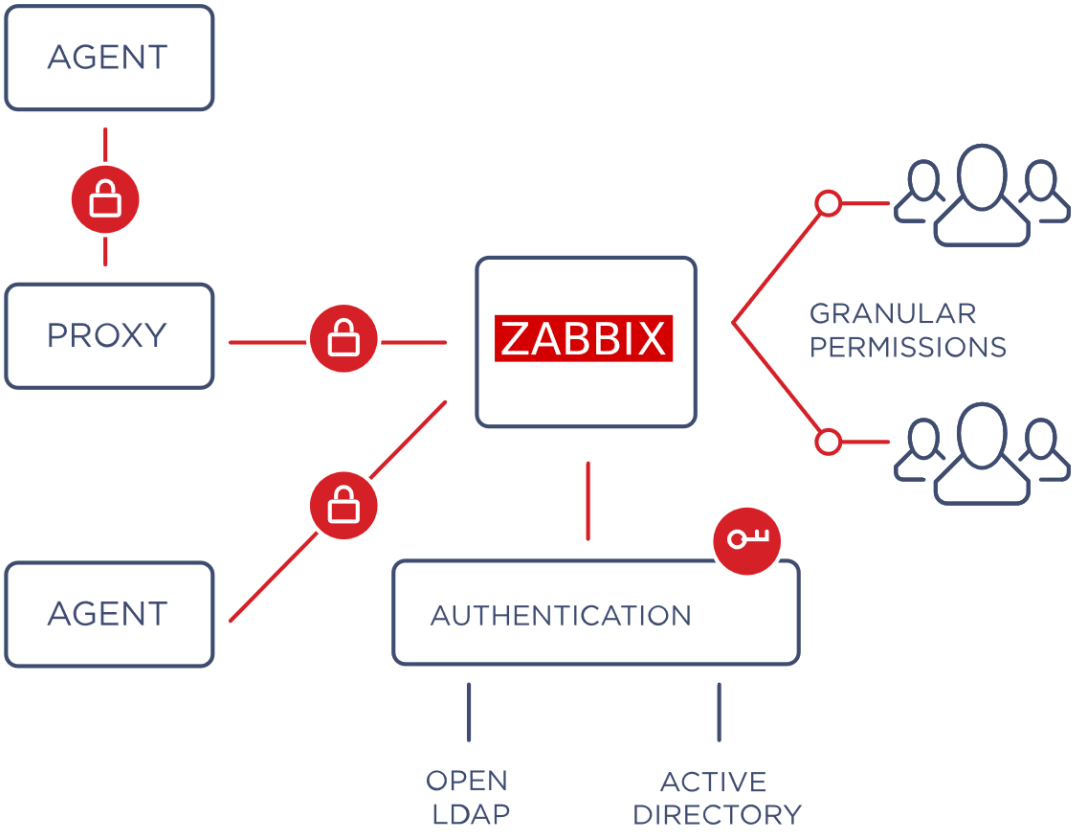
2 KÄYTETYT TEKNOLOGIAT

2.1 Zabbix

Zabbix on yrityskäyttöön suunnattu avoimen lähdekoodin järjestelmävalvonta ohjelmisto, jonka ensimmäinen versio julkaistiin 2001. Zabbix:n tavoitteena on tehdä kaikkiin ympäristöihin sopiva valvontaratkaisu, joka on myös kaikille saatavilla edulliseen hintaan. (Zabbix About Us, 2018). Zabbix on käytettävissä ilmaiseksi, myös kaupallisesti (Zabbix Features, 2018).

2.1.1 Toiminta

Zabbix:n toiminta perustuu lähtökohtaisesti yhteen keskitettyyn Zabbix palvelimeen, joka kerää kaikkien valvottavien järjestelmien tiedot Zabbix agenttien kautta (kts KUVA 1 alla) (Zabbix Features, 2018). Zabbix agentti on valvottavaan järjestelmään asennettava ohjelmisto, joka kerää kaiken kyseisestä järjestelmästä halutun tiedon ja lähettää sen eteenpäin. Zabbix tukee myös monia muitakin valvonta keinoja, kuten esim. SNMP, IPMI ja SSH, mutta Zabbix agentti on pääasiallinen valvonta tapa. Kerättiin tiedot millä keinolla hyvänsä, tiedot voidaan lähettää joko suoraan Zabbix palvelimelle tai välityspalvelimen kautta (Zabbix proxy). Zabbix välityspalvelin on käytännössä sama kuin perus Zabbix palvelin paitsi että välityspalvelimessa ei ole web käyttöliittymää. Toisin sanoen välityspalvelin on lähinnä tiedon väliaikaista säilytystä, eteenpäin välitystä ja palvelimen kuorman tasapainottamista, eikä varsinaista valvontaa varten.



KUVA 1. Zabbix toiminta periaate. *(Zabbix Features, 2018).*

Käytännön syitä välityspalvelimen käytölle on esimerkiksi verkon yksinkertaistaminen. Mikäli halutaan valvoa esimerkiksi palomuurin takana olevaa pilvipalvelinta. Pilvipalvelimessa virtuaalikoneita tulee ja menee, joten sen sijaan että yritetään yhdistää joka virtuaalikone palomuurin lävitse, asennetaan Zabbix välityspalvelin pilven sisälle, jonka liikenne ohjataan palomuurin lävitse varsinaiselle Zabbix palvelimelle. Näin kaikki virtuaalikoneet voivat yhdistää pilven sisällä olevalle välityspalvelimelle ilman että tarvitsee määritellä virtuaalikoneille palomuuri sääntöjä. Toinen syy on esimerkiksi historia tietojen säilymisen varmistamiseksi, koska Zabbix voi valvoa järjestelmiä verkon välityksellä voivat järjestelmät sijaita, vaikka toisella mantereella. Zabbix agentit voivat kyllä lähettää tietonsa suoraan Zabbix palvelimelle mutta jos yhteys katkeaa, kaikki katkoksen aikainen data menetetään, ellei agenttien data mene välityspalvelimen kautta, jolloin katkoksen sattuessa Zabbix välityspalvelin voi säilyttää katkoksen aikana kerätyn datan, kunnes yhteys taas muodostetaan, jolloin mitään kerättyä dataa ei menetetä.

2.1.2 Järjestelmä vaatimukset

Zabbix on hyvin eri käyttötarkoituksiin mukautuva ohjelmisto, joka toimii pienessäkin ympäristössä hyvinkin vähillä resursseilla. Alla, taulukossa 1, näkyy Zabbix:n suurpiirteiset järjestelmä vaatimukset muistin, prosessorin sekä tietokannan osalta.

TAULUKKO 1. Zabbix järjestelmä vaatimukset. (Zabbix System Requirements, 2018).

Name	Platform	CPU/Memory	Database	Monitored hosts
Small	CentOS	Virtual Appliance	MySQL InnoDB	100
Medium	CentOS	2 CPU cores/2GB	MySQL InnoDB	500
Large	RedHat Enterprise Linux	4 CPU cores/8GB	RAID10 MySQL InnoDB or PostgreSQL	>1000
Very large	RedHat Enterprise Linux	8 CPU cores/16GB	Fast RAID10 MySQL InnoDB or PostgreSQL	>10000

Käytännössä vaatimukset kuitenkin riippuvat suuresti itse valvottavista järjestelmistä, kuten montaako järjestelmää valvotaan, kuinka paljon dataa niistä kerätään ja kauanko niitä halutaan säilyttää. Varsinkin levytilan osalta nämä ovat tärkeitä osata arvioida, kun tiettyjä tietoja voidaan joutua säilyttämään, laista riippuen, jopa vuosia.

Zabbix tukee myös monia eri alustoja, kuten Linux, Mac OS X, Windows (XP tai vanhempi) sekä uaeita muita. Huomioitavaa on, että kaikki alustat paitsi Windows tukee sekä palvelimia että agentteja, Windows nimittäin tukee pelkästään agenttia. (Zabbix Platforms, 2018).

2.1.3 Käyttöliittymä

Zabbix on melko vanha ohjelmisto kilpailijoihinsa nähden, mikä näkyy lähinnä käyttöliittymässä, se kun ei ole se kaikista modernein. Tosin sen puutteen paikkaa kolmannen osapuolen ohjelmistot, kuten vaikka Grafana jolle on Zabbix liitännäinen millä saa rakennettua hyvinkin modernin näköisen käyttöliittymän. Tosin tässä opinnäytetyössä käyttöliittymä puoli ei ole pääasia.

2.1.4 Ominaisuudet

Zabbix pystyy valvomaan monipuolisia järjestelmiä, laitteistoja, sovelluksia, tietokantoja, verkkosivuja, pilvi ympäristöjä ja kontteja. Kaikille näille voidaan määrittä tiettyjä raja-arvoja, joiden mukaan annetaan hälytyksiä, tehdään ilmoituksia (sähköposti, tekstiviesti tai Jabber- chat viesti) ja/tai suoraan ryhdytään toimenpiteisiin. (Zabbix Features, 2018)

Zabbix pystyy seuraamaan datan trendejä ja huomaamaan poikkeavuuksia, esim. prosessorin käyttö tiettyyn aikaan poikkeaa aikaisemmista päivistä tai ennustamaan ongelma tilanteita, esim. jos levytila vähenee tasaisesti, voidaan ennustaa koska se loppuu ja ilmoittaa siitä etukäteen. (Zabbix Features, 2018).

2.2 Python

Python on Guido van Rossum:in 90-luvulla kehittämä ohjelmistokieli joka on myös avointa lähdekoodia (python.org License, 2018). Pythonin tärkeimmät ominaisuudet ovat sen selkeä ja helppolukuinen syntaksi sekä iso kirjasto jolla pystyy nopeasti toteuttamaan monia oikeasti hyödyllisiä sovelluksia. Pythonin ominaisuuksista johtuen sitä pidetään myös hyvänä ohjelmisto kielenä ensikertalaisille. (python.org Doc, 2018)

Otetaan esimerkiksi perinteinen ”Hello world” sovellus, kuviossa 1 näkyy ”Hello world” toteutettuna sekä C++:lla että Pythonilla. Kuten kuviosta näkyy, on Pythonin syntaksi huomattavasti lyhyempi ja selkeämpi, eikä se sisällä mitään erikoismerkkejä. (python.org Doc, 2018).

```
C++
#include <iostream>
int main()
{
    std::cout << "Hello World!";
}

Python
print "Hello, World!"
```

KUVIO 1. Pythonin ja C++ vertailu.

Nykyään Python on myös suosittu ohjelmistokieli koneoppimiseen, esimerkkinä mainittakoon Googlen avoimen lähdekoodin koneoppimis kehys ”Tensorflow” (towardsdatascience.com, 2018).

2.3 Unix Shell Bash

Bash on komentorivitulkki ja hyvin suosittu Linux ympäristöissä, jonka takia se tulee monessa Linux käyttöjärjestelmässä vakiona. Sen suurimpia ominaisuuksia edeltäjiinsä nähden ovat sen vuorovaikutteisuus, komento historia, komentorivi muokkaus ja aliakset. (Bash FAQ, 2018).

2.4 jq

Jq on komentorivi työkalu, jota käytetään esim. Bash:in kanssa. Jq:lla voidaan suodattaa ja muokata JSON rakenteita hyvin vaivattomasti. Toteutettu C kielellä ilman riippuvuuksia. (jq, 2018).

Syntaksi on hieman vaikea oppia, mutta kun sen ymmärtää on jq erittäin tehokas työkalu JSON käsittelyyn. Jq toimii siten että jokaiselle suodattimelle on yksi sisäänmeno ja yksi ulostulo. Otetaan esimerkiksi taulukko [1,5,3,0,7], ja sanotaan vaikka että haluamme tehdä luvuista listan JSON objekteja, mutta vain niistä luvuista jotka ovat suurempia kuin kaksi. Tämä onnistuu taulukon 2 ensimmäisellä rivillä olevalla jq suodattimella. Taulukossa riveillä 2-4 selventäviä välivaiheita.

TAULUKKO 2. Jq suodattimia.

Sisään	**jq suodatin**	**Ulos**
[1,5,3,0,7]	map(select(. > 2)) | .[] | [foreach . as $arvo (.;{"Luku": $arvo})]	[{"Luku": 5}, {"Luku": 3}, {"Luku": 7}]
[1,5,3,0,7]	map(select(. > 2))	[5,3,7]
[5,3,7]	.[]	5 3 7
5 3 7	[foreach . as $arvo (. ;{"Luku": $arvo})]	[{"Luku": 5}, {"Luku": 3}, {"Luku": 7}]

Jq syntaksissa ”.” tarkoittaa sisäänmenoa sellaisenaan, ”|” tarkoittaa että edellisen suodattimen (”|” vasemmalla puolella) tulos syötetään seuraavaan suodattimeen (”|” oikealla puolella).

Taulukon 2 ensimmäisellä rivillä koko suodatin jolla saadaan haluttu tulos. Rivillä 2 olevassa välivaiheessa suodatetaan pois luvut jotka ovat pienempiä kuin 2, rivillä 3 jäljelle jääneet luvut otetaan pois taulukosta ja asetellaan omille riveille, rivillä 4 joka rivi muotoillaan JSON rakenteeksi ja koska koko jq suodatin on hakasulkeiden sisällä, laitetaan sen ulostulo taulukon sisään.

2.5 MQTT

MQTT on erittäin yksinkertainen ja kevyt viestintä protokolla jolla haluttiin saada pienitehoisille laitteille sopiva protokolla, joka olisi myös luotettava. Tämä on tehnyt siitä myös hyvän vaihtoehdon Internet of Things laitteille. (mqtt.org, 2018).

MQTT toiminta perustuu viestien julkaisuun ja tilaukseen, tarkoittaen että MQTT palvelimelle voi julkaista viestejä tietyllä otsikolla, mikä mahdollistaa tilauksen

tekemisen kyseisellä otsikolla, jolloin kaikki kyseiselle otsikolle julkaistut viestit lähetetään kaikille otsikon tilaajille. Tämä tarkoittaa, ettei julkaisijoiden ja tilaajien tarvitse tietää toistensa olemassa olosta, kuten IP osoitteita yms., vaan riittää että kaikki tietävät MQTT palvelimen osoitteen ja halutun otsikon. (mqtt.org, 2018).

MQTT protokollan kehittivät vuonna 1999 Andy Stanford-Clark IBM:ltä, sekä Arlen Nipper Arcom:lta. (mqtt.org, 2018). Nykyään MQTT on virallinen ISO standardisoitu (ISO/IEC 20922) viestintä protokolla (International Organization for Standardization, 2016).

2.6 Teknologioiden valinnat

Teknologia valinnat pohjautuivat projektitiimille tutuista teknologioista, tosin tämän projektin vaatimukset jonkun verran rajoittavat teknologia vaihtoehtoja.

Zabbix:lle löytyy monia kilpailijoita, tosin sitä on tiimin sisällä käytetty muissakin projekteissa, ja on hyväksi todettu, jo ennen minun saapumista taloon, joten sitä käytettiin tässäkin. Muita teknologia valintoja yhdistää avoin lähdekoodi ja vapaa käyttö myös kaupallisiin tarkoituksiin.

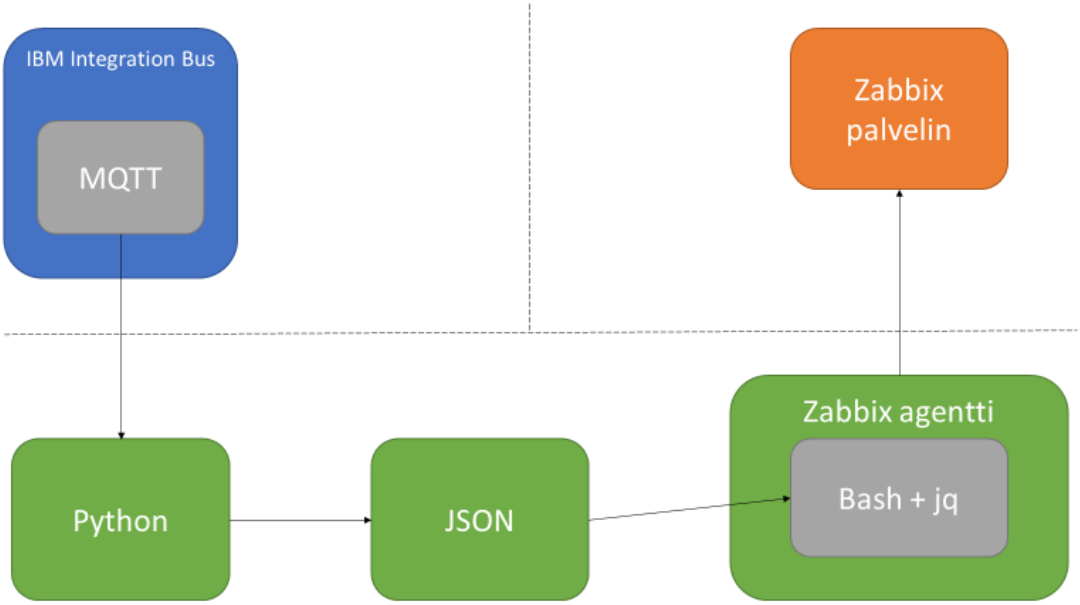
3 TOTEUTUS

3.1 Tavoite

Tämän projektin tavoitteena oli siis toteuttaa sovellustason valvonta IBM Integration Bus:lle Zabbix:ssa. Ennakkoon IIB:stä tiedettiin, että sen tapahtumiin päästään käsiksi sisäisen MQTT palvelimen kautta.

3.2 Looginen arkkitehtuuri

MQTT palvelimen kuuntelu päätettiin toteuttaa python skriptillä, jonka pääasiallinen tehtävä on vastaanottaa MQTT palvelimen viestejä ja kirjoittaa ne levylle JSON tiedostoon. Kun tiedot on saatu levylle, voidaan niitä kysellä Zabbix agentin käyttäjä parametreja hyödyntäen. Kyselyt toteutetaan erinäisillä Bash skripteillä jotka suodattavat halutun tiedon jq avulla. Jokainen Bash skripti hakee tietyn tyyppistä dataa, kuten integration node:n tilaa tai sen messageflow:n tietoja. Lopuksi Zabbix agentti lähettää tiedot Zabbix palvelimelle jossa agentin keräämälle datalle voidaan määrittää valvontaan tarpeelliset hälytykset. Arkkitehtuuri visualisoitu kuviossa 2.

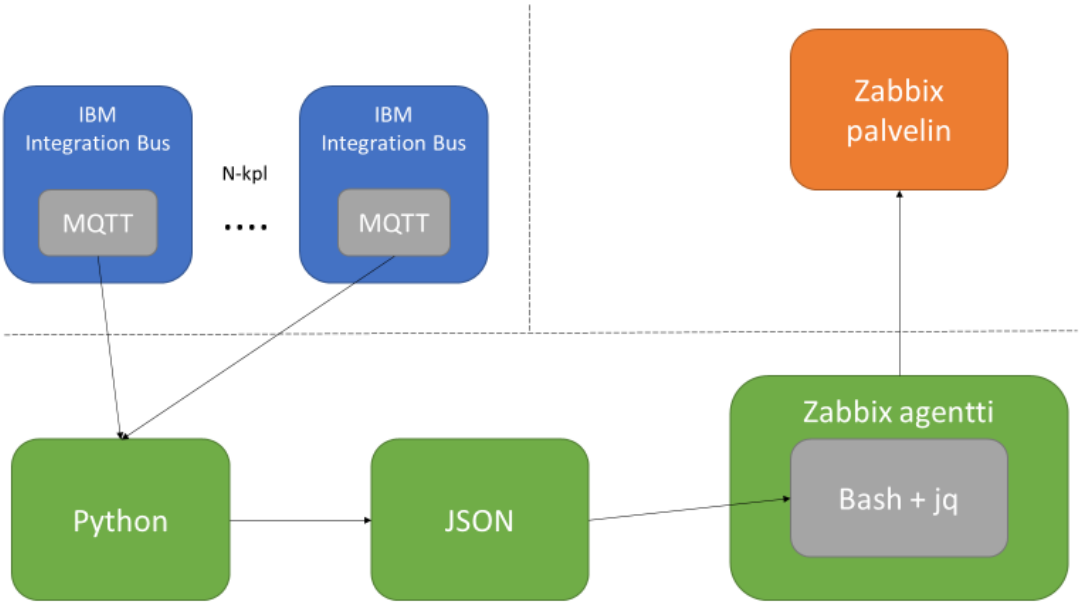


KUVIO 2. Yksinkertaistettu looginen arkkitehtuuri.

Python skripti on toteutettu siten että sitä ei tarvitse ajaa paikallisesti IBM Integration Bus kanssa, vaan sillä voidaan vastaanottaa viestejä myös verkon yli. Tämä tulee olemaan tarpeen erityisesti jatkokehitystä ajatellen josta enemmän seuraavassa kappaleessa.

4 JATKOKEHITYS JA POHDINTA

Projektin alkuperäiset tavoitteet saavutettiin onnistuneesti, tosin kuten projekteissa usein käy, muuttuivat tavoitteet projektin edetessä. Tässä tapauksessa huomattiin tarve useamman IIB MQTT palvelimen saman aikaiseen valvomiseen, mikä tarkoittaa Python skriptin MQTT kuuntelun säikeistämistä ja muuutenkin uudelleen kirjoittamista. Jaktokehitys visualisoitu kuviossa 3.



KUVIO 3. Jatkokehityksen looginen arkkitehtuuri.

Tämän opinnäytetyön kirjoitus hetkellä Python skriptin muutokset ovat melko pitkällä, ja testiympäristössä jossakin määrin toimivia, tosin se on vielä melko testaamaton. Valvontaratkaisua on tarkoitus vielä testata oikeassa tuotanto ympäristössä, jonka aikana saadaan tarkempaa tietoa tämän valvontaratkaisun tarpeista.

Projekti itsessään meni oikein hyvin, tosin muutoksien totetuttamiseen olisi vain tarvittu enemmän aikaa, mutta kehitys jaktuu töiden merkeissä. Teknologiat olivat minulle osittain tuttuja ja osittain täysin tuntemattomia, tekemistä helpotti huomattavasti se että olin tutustunut Zabbixin käyttöön jo edellisenä kesänä Nokialla. Tosin vaikka Zabbix enensetään tuttu olikin, oli tälle projektille tärkeimmät Zabbixin ominaisuudet minulle uusia. Pythonia en ollut käytännössä käyttänyt juuri lainkaan, mutta Python on tunnetusti helppo oppia joten siinä ei juuri yllätyksiä tullut. Suurin haaste tässä oli pikemminkin itse

kokonaisuus ja kaikkien sen osien liittäminen yhteen. Päänvaivaa myös aiheutti Jq jonka syntaksi oli alkuun hyvin vaikea selkoinen, vaikkakin tehokas.

LÄHTEET

Bash FAQ. (21. Toukokuu 2018). Noudettu osoitteesta ftp://ftp.cwru.edu/pub/bash/FAQ

International Organization for Standardization. (Kesäkuu 2016). Viitattu 23.5.2018. Noudettu osoitteesta https://www.iso.org/standard/69466.html

jq. (21. Toukokuu 2018). Noudettu osoitteesta https://stedolan.github.io/jq/

mqtt.org. (23. Toukokuu 2018). *Frequently Asked Questions*. Noudettu osoitteesta http://mqtt.org/faq

python.org Doc. (25. Toukokuu 2018). *Is Python a good language for beginning programmers?* Noudettu osoitteesta https://docs.python.org/3/faq/general.html#is-python-a-good-language-for-beginning-programmers

python.org License. (25. Toukokuu 2018). *History and License*. Noudettu osoitteesta https://docs.python.org/3/license.html

towardsdatascience.com. (25. Toukokuu 2018). Noudettu osoitteesta https://towardsdatascience.com/what-is-the-best-programming-language-for-machine-learning-a745c156d6b7

Zabbix About Us. (4. Toukokuu 2018). Noudettu osoitteesta https://www.zabbix.com/about

Zabbix Features. (4. Toukokuu 2018). Noudettu osoitteesta https://www.zabbix.com/features

Zabbix Platforms. (17. Toukokuu 2018). Noudettu osoitteesta https://www.zabbix.com/requirements

Zabbix System Requirements. (17. Toukokuu 2018). Noudettu osoitteesta https://www.zabbix.com/documentation/3.4/manual/installation/requirements

LIITTEET

Liite 1. Linkki projektin GitHub sivulle

https://github.com/digiapulssi/zabbix-iib-monitoring